

MODERNIZING MINECRAFT

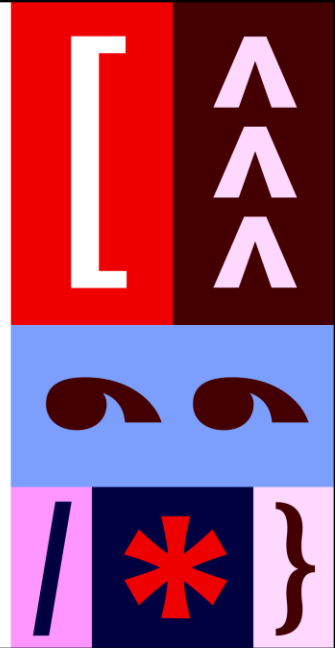
Evolving the Rendering Pipeline

A.J. Fairfield (he/him)

Mojang

#GDC2026

Classified as Microsoft Confidential



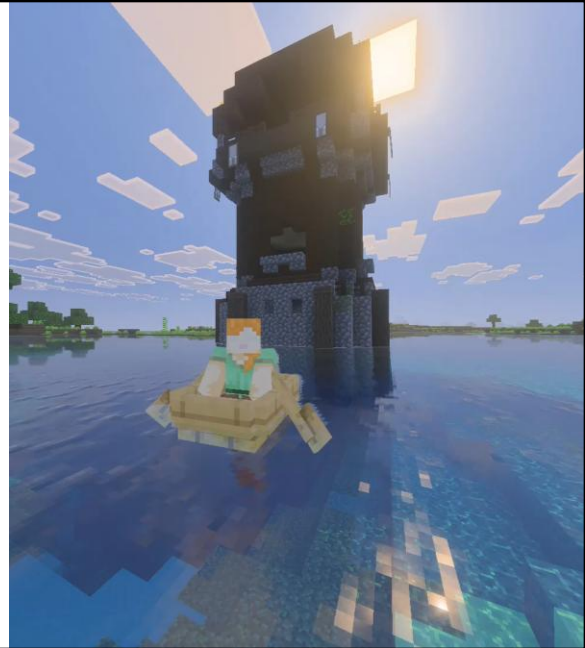
AGENDA

- 1 Rendering Journey of Minecraft
- 2 Traditional Minecraft Lighting 101
- 3 Dive into Vibrant Visuals: our brand-new, cross-platform, real-time rendering pipeline released in June 2025
- 4 Performance Statistics and Targets
- 5 Community Driven Development

[GDC] Festival of Gaming | MARCH 9 - 13, 2026
SAN FRANCISCO, CA

#GDC2026

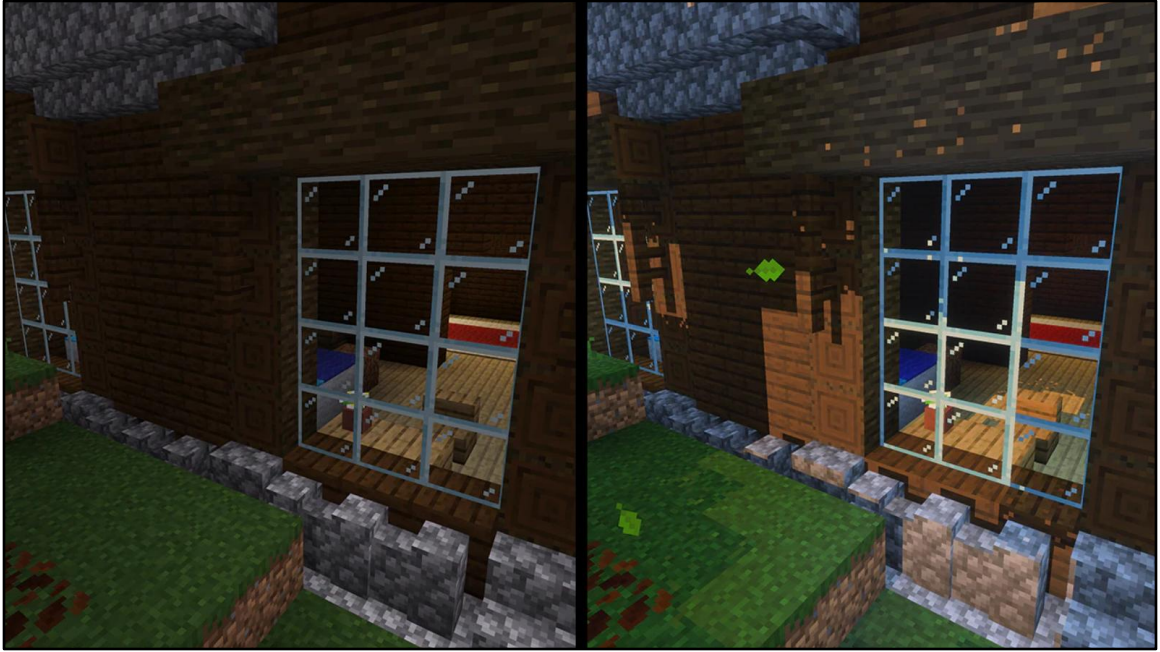
Classified as Microsoft Confidential



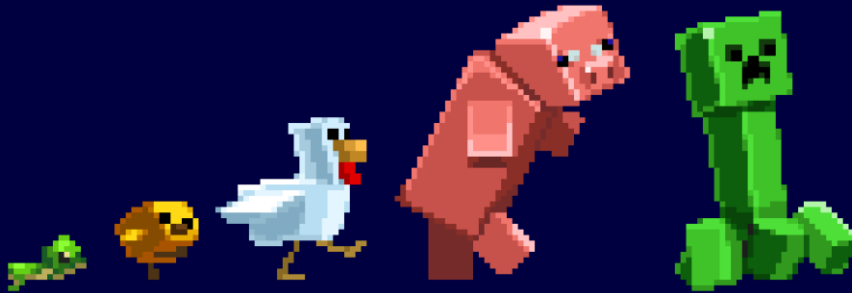


<https://youtu.be/jxvDnhZ7WXw>





RENDERING JOURNEY



How Minecraft's rendering has evolved over time

RENDERING JOURNEY

Cross Platform Requirements

Because we all know that Minecraft runs on a toaster

13 Unique Platforms

Minecraft Java Edition

Linux, MacOS and Windows

Minecraft Bedrock Edition

Mobile: Amazon, Android, Chromebook, iOS

Console: Switch, PS4, PS5, XBOX One and XBOX Series

PC: MacOS (EDU Edition only) and Windows

8 Graphics APIs (12+ shader models)

D3D11, D3D12, OpenGL, GLES, PS4, PS5, Vulkan, Metal

Multiple HW generations and IHVs

Game uses render distance to scale across full gamut of HW

Android Min Spec: OS 9 + 1GB RAM + GLES 3.1

RENDERING JOURNEY

The Early Years

2009



Java Renderer

Immediate mode rendering tightly coupled with gameplay

OpenGL only

Minecraft Java

2011



Minecraft HAL

Graphics API abstraction for Bedrock's cross-platform needs; still immediate mode

D3D11, GLES 2

Minecraft Pocket Edition

2016



Minecraft HAL/IR

First attempt at supporting "modern" graphics APIs; limited by existing HAL interface

D3D12, Switch

Minecraft Bedrock

 **Festival of Gaming** | MARCH 9 - 13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential

RENDERING JOURNEY

Super Duper Graphics Pack

Developed on the D3D12 backend for XBOX One S and X first
Intended to ship on all platforms

Ultimately canceled, but why?

1. Didn't use a PBR model
2. Immediate mode rendering architecture was too limiting

So, what were we going to do now?

[GDC] Festival
of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026



Why was it cancelled from a tech perspective?

- 1) We didn't use a PBR model, so we could never quite get all times of day or every environment tuned properly. Being that Minecraft's world is fully deformable, we must be resilient to lights being placed anywhere and terrain changing at a moments notice. By not focusing on energy conservation, we ended up with systems that fought against each-other and created headaches for our artists, in some cases causing insurmountable problems.
- 2) On a completely different vein, we also didn't realize just how much the immediate-mode rendering nature of the existing code would limit us and cause massive performance problems. We were trying to do simple things like "draw the world from a different perspective" for generating shadow maps let's say, but we quickly hit walls that made even the simplest techniques unfeasible.

RENDERING JOURNEY



*The RenderDragon Logo
By Andy Hill*

Enter the RenderDragon

Modern, platform-agnostic rendering framework

Sits on top of a HAL library, BGFX in our case

Key Principles

Heavily inspired by Natalya Tatarchuk's GDC 2015 Destiny talk

Decouples gameplay logic from drawcalls via a frame registry

Resource handles are tracked in a client/server model

Games provide frame graph structure

Shaders should be easy to write

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential

We introduced a hard hand-off layer between gameplay and rendering code where the game would make requests to draw an object. We do a copy of all necessary gameplay data to create said draw call, and we respect the intent of the game play submission, but when that draw actually occurs is ultimately up to us. This copy of data resides in a lightweight database called a frame registry that we recreate every frame.

We further decouple the game from graphics by handing out all GPU resource handles for things like buffers or textures in a client/server style model, where the game is the client and RenderDragon is the server. This means the game just holds onto opaque handles that they can use to represent the GPU resource without having to worry about details of lifetime or read/write barriers. This has helped us tremendously for app lifetime quirks of different platforms, such as suspend/resume.

The game still tells us how to assemble “a frame”, but we provide an API for consuming the frame registry and deciding how to present it.

RENDERING JOURNEY

DragonFX

Cross platform shader language

Custom built using ANTLR

Key Principles

Write once, run anywhere

All compilation happens outside of the game

- Improved iteration time
- Higher optimization potential
- Service model allows the game to discover material updates to facilitate hot reloading of shaders

Leverages DXC for HLSL and SPIRV-Cross for Vulkan and Metal

```
// Copyright (c) Rejion AB. All rights reserved.

MaterialDefinition "RenderChunk"
{
    Properties
    {
        [Color] FogColor: #ffffff
        [vector] FogDistanceControl: (0, 0, 0, 0) // x = min fog range, y = max fog range, z = render distance
        [vector] RenderChunkFogAlpha: (0, 0, 0, 0)
        [Sampler2D] LightMapTexture
    }
    $import "RenderChunkUniforms.dragon"
}

Shared
{
    Declaration
    {
        [Flag] Instancing : Off, On
        [Flag] Seasons : Off, On
        [Flag] RenderAsBillboards: Off, On
        [Flag:Invert] [Seasons:On, RenderAsBillboards:On]

        [Option] lighting : RenderChunkLighting

        [VertexInput:vec3] position : POSITION
        [FragmentInput:vec2, Interpolation:Centroid] texcoord0 : TEXCOORD0
    }
    Code
    {
        #include "TARUill.dragon"
        #include "Fogutil.dragon"
        #include "RenderChunkBill.dragon"
        #include "C3STerrainIntrinsics.dragon"

        float RenderChunkVert(StandardVertexInput stdInput, inout VertexOutput vertOutput) {
            #if !flag(RenderAsBillboards, On)
                vertOutput.color = vec4(1.0, 1.0, 1.0, 1.0);
                transformAsBillboardVertex(stdInput, vertOutput);
            #endif
        }
    }
}
```

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential

DragonFX is what we call our material language. It's a custom decoration on top of GLSL that we maintain with our own ANTLR grammar. We have decorations for things like passes, properties, shared code sections, etc...

RENDERING JOURNEY

Post RenderDragon Timeline

2017



RenderDragon

First used in Minecraft Earth; rolled out over time on Bedrock

D3D12, D3D11, Metal, Vulkan, GLES, PS4

Minecraft Earth, Minecraft Bedrock

2018



"Look! I'm a real engine!"

Battle testing of RenderDragon; not limited by HAL design issues

D3D12, Switch, PS4

Minecraft Legends

2019



Minecraft RTX

Partnership with Nvidia; fully path-traced w/ PBR art assets

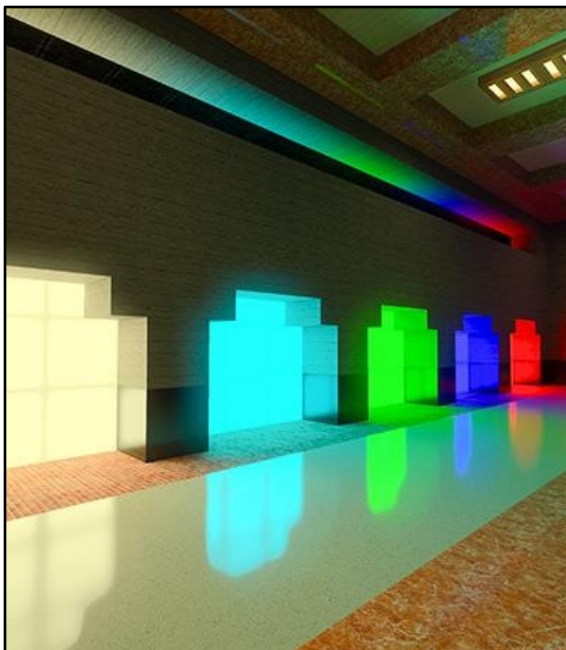
D3D12 PC Only; DXR 1.1

Minecraft Bedrock + DXR

 Festival of Gaming | MARCH 9 - 13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential



RENDERING JOURNEY

The decision to make Vibrant Visuals

RTX and Java made it clear: players wanted better graphics

We wanted to embrace cross-platform accessibility

Deferred Rendering Pipeline was the answer

Plenty of tech to utilize from Minecraft Legends

- DragonFX shading modules for an HDRi pipeline

Leverage some components of Minecraft RTX

- Asset pipeline for PBR material data
- G-buffer populated for the first bounce of path tracing

GDC Festival
of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

MINECRAFT LIGHTING 101



Establishing the basics of
lighting in Minecraft

[GDC] Festival
of Gaming | MARCH 9 - 13, 2026
SAN FRANCISCO, CA

#GDC2026

Powered by Microsoft Commercial

MINECRAFT LIGHTING 101

Terrain Assembly

World is divided into "chunks" of 16x16x16 blocks

Chunks are queued for assembly as the player moves around

Graphics Details

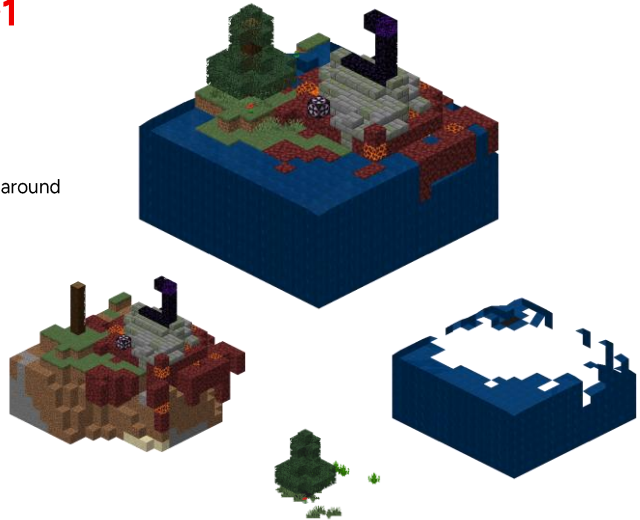
Vertex pool is pre-allocated in pages

Single VB range assigned per chunk

Multiple IB ranges per terrain layer

E.g., opaque, foliage, water, blended

Lighting is baked into each vertex



[GDC] Festival of Gaming

MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential



MINECRAFT LIGHTING 101

Sky Lighting

Measure of sky exposure (0-15)
Heightmap based evaluation

Graphics Details

Allows light to bleed around corners up to 15 steps
Data is more representative of an indirect term than direct
Lighting is baked and repropagated as the terrain changes
Normalized value stored in each vertex

[GDC] Festival of Gaming | MARCH 9-13, 2006
SAN FRANCISCO, CA

#GDC2026

Lighting in Minecraft consists of 2 pieces, the first one being sky lighting.

This is essentially a measure of exposure to the sky, represented with a value between 0-15, and it's imagining that light is illuminating from the sky uniformly. It is using a heightmap-based evaluation of sky visibility, which can break down in certain cases. For instance, you can see the areas under the overhangs in this screenshot here. Even when the sun is at sunset at an oblique angle, the blocks underneath the overhangs will still have just as much sky exposure as they do at noon. Even with this limitation, the system works pretty well for caves and buildings.

MINECRAFT LIGHTING 101

Block Lighting

Measure of light intensity from block sources (0-15)

Monochromatic lighting

Graphics Details

Uses Manhattan/Taxicab distance metric instead of Euclidean

$$distance(a, b) = \sum_{i=1}^n |a_i - b_i|$$

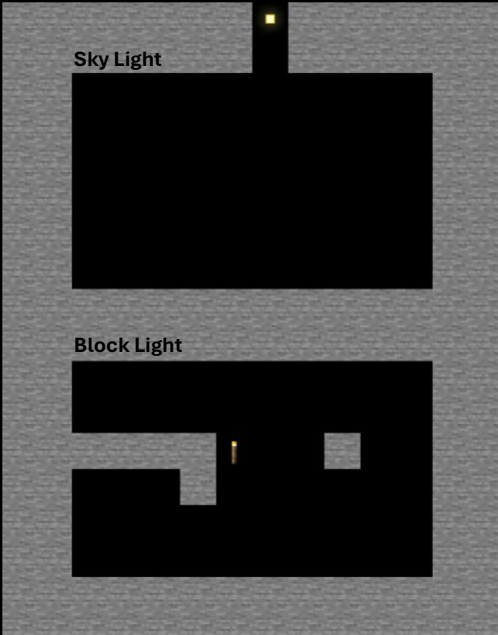
Allows light to bleed around corners up to 15 steps

Data represents a combo of both direct and indirect terms

Lighting is baked and repropagated as the terrain changes

Normalized value stored in each vertex





Sky Light

Block Light

MINECRAFT LIGHTING 101

Light Propagation

Multiple rounds of breadth-first, flood-fill propagation
Lighting falloff is linear and can bleed around corners

Algorithm Details

Seeding or initializing the volume:

- Mark occluders (e.g., walls) and absorbers (e.g., water)
- Block and sky light values initialized based on source block positions and direct sky exposure respectively

Propagating or flood-filling the light:

- Multiple waves based on largest light intensity
- Neighbor visitation strategy allows light to wrap corners
- Propagate block and sky light as separate terms

 Festival of Gaming

MARCH 9-13, 2006
SAN FRANCISCO, CA

#GDC2026

Both block and sky lighting utilize a CPU-side based method of light propagation that's carried out in multiple rounds of breadth-first, flood-fill exploration. Think Dijkstra's graph algorithm. And the lighting attenuation is simply linear in that light falls off at a rate of 1 level of intensity per block.

At a high-level, the algorithm breaks into 2 parts.

- 1) First, we seed a light volume with our light sources and various occluders like walls or absorbers. Water is an absorber of light in our model, for instance, whereas an opaque block is an impenetrable wall for light. Sky lighting is seeded by setting any cell at or above the heightmap to a value of full sky light.
- 2) Next, we run multiple steps of propagation based on the largest light intensity in the region of blocks we are updating. The flood fill aspect of our discovery checks adjacent neighbors for further propagation, and this is what allows light to bleed around corners up to 15 steps. We keep track of block and sky light as separate terms.

MINECRAFT LIGHTING 101

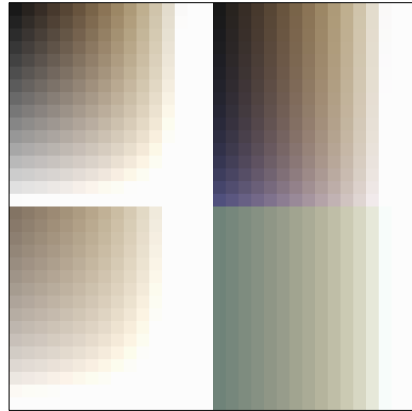
Brightness Texture

Dynamically generated and updated at runtime

- Different times of day
- Different dimensions

Sky and block light values are used to lookup from a texture

- X-axis: block light
- Y-axis: sky light



Once we have sky and block light values, we need to turn them into colors. We do this with a simple texture lookup.



MINECRAFT LIGHTING 101

Ambient Occlusion

Cheap approximation of shadowing in crevices

Gives a crucial sense of depth to the scene

Algorithm Details

For each face

- Look up neighboring blocks
- Calculate light level, ll , for each vertex

$$B(block) = \begin{cases} 1.0; & \text{transparent blocks} \\ 0.2; & \text{solid and leaf blocks} \end{cases}$$

$$ll = \frac{B(facingDir) + B(side1) + B(side2) + B(corner)}{4}$$

- Tint base color by computed AO light level

GDC Festival of Gaming | MARCH 9-13, 2006
SAN FRANCISCO, CA

#GDC2026

MINECRAFT LIGHTING 101

Terrain Atlas

Mega-texture with all the block faces

Nothing too fancy

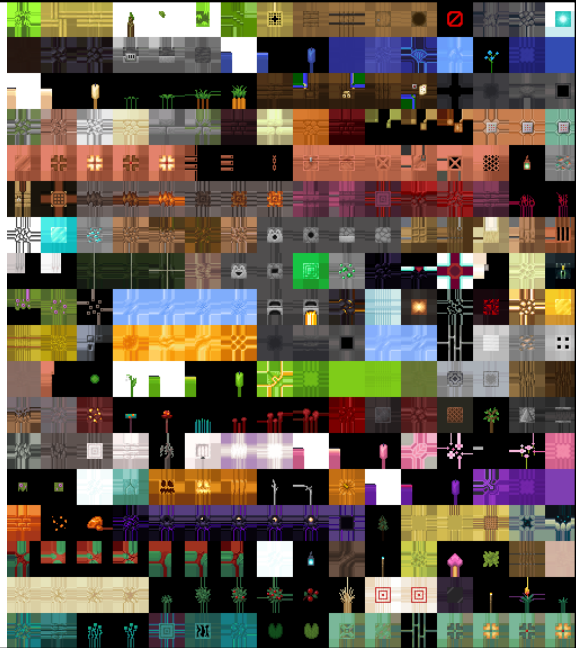
Details

Contains color only

Vertex UVs correspond to atlas tiles

Tiling supports face variations and rotations for randomness

Animated block faces are just subregion updates to the texture



[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential

VIBRANT VISUALS



Our brand-new rendering pipeline.
So sparkly. Much wow.

[GDC] Festival
of Gaming | MARCH 9 - 13, 2026
SAN FRANCISCO, CA

Classified as Microsoft Confidential

#GDC2026

VIBRANT VISUALS

Material Model

Albedo, Normals, Metalness, Emissive, Roughness, Subsurface

Details

Use existing color assets for "Albedo"

We allow for partial metals, but subsurface and metallics are mutually exclusive

All attributes can be specified uniformly across entire faces or painted via textures

No programmable index of refraction (would be useful for gems and ore), clearcoat, sheen, or AO

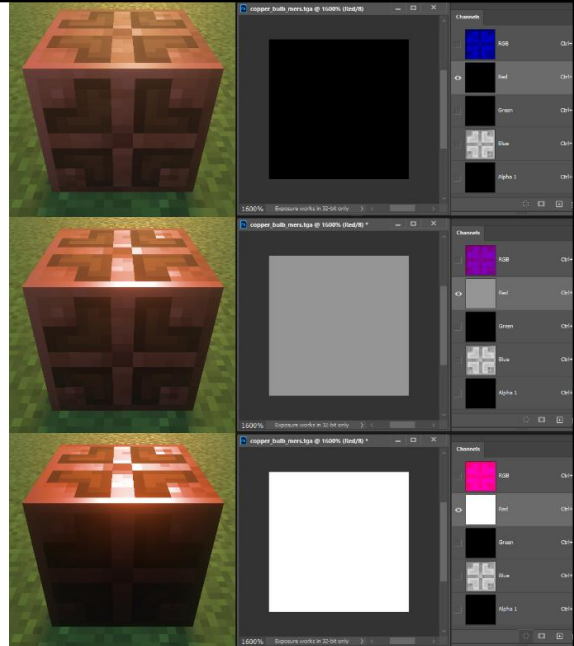
Ended up producing over 3,000 textures as part of this effort

- Automated art-gen via material presets and masks

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential



We allow for partial metals, which isn't typical, but we felt that Minecraft shouldn't be constrained by the same rules as materials we've discovered in our universe, so we will calculate both dielectric and conductive variants and blend them based on the metalness factor. Subsurface and metalness are mutually exclusive in our model though. This is a stylistic choice, but also a storage choice for memory optimization purposes.

All and all, we ended up making over 3,000 new textures. We accomplished this volume in a very short time by getting the Art team to really embrace PBR. It didn't take much to get them to start reimagining all of the Minecraft blocks in terms of more than just color. First, they spent time classifying all of the various materials in the game: stones, leathers, ores, etc... They then built a hierarchy of blocks and what materials they share, because some blocks contain multiple material types, and paired each one of those with a material mask. An automated process takes those material definitions and masks and generates all of our non-color assets. This allowed us to iterate quickly and regenerate all art assets with a single click. You can autogen more than just code. 😊

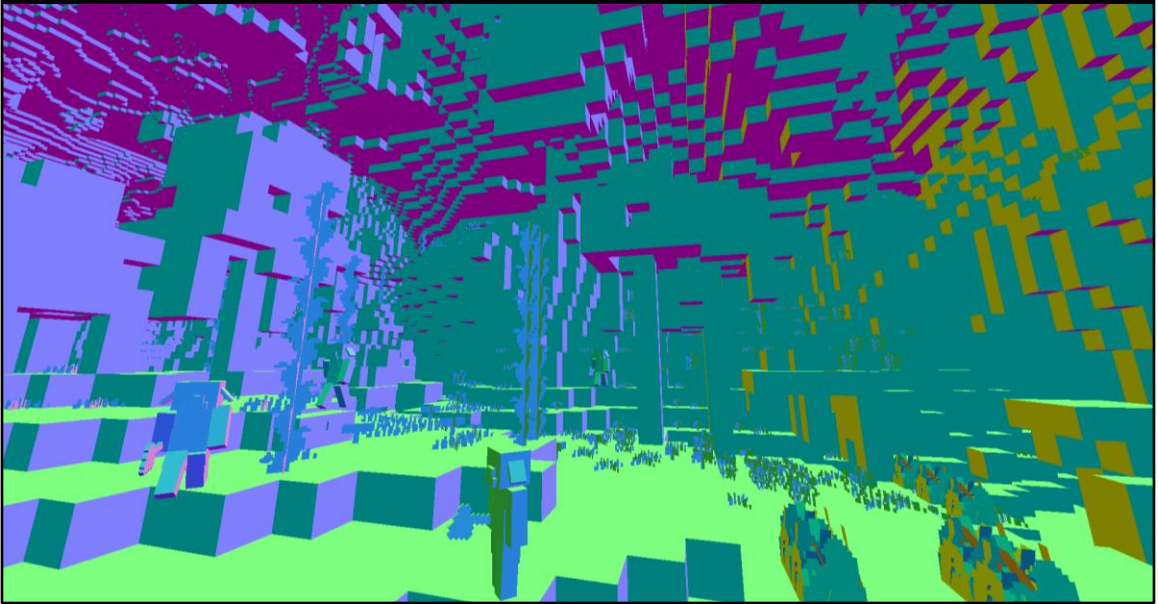
Final Shaded Scene



Albedo



Normals



Roughness





You can see little touches of metal scattered throughout the scene. On the belt buckles, some bits on the treasure chest and even some ore up in the cavern ceiling.



Final Shaded Scene



VIBRANT VISUALS

Gameplay Considerations for Lighting

Must preserve all gameplay cues that are lighting based

Players truly depend on light mechanics

PBR Constraints

Time of Day: sun and moon dictates when player can sleep or when hostile mobs will spawn in the open

Block Lights: create safe areas where mobs will not spawn and allows exploration of dark areas such as mineshafts

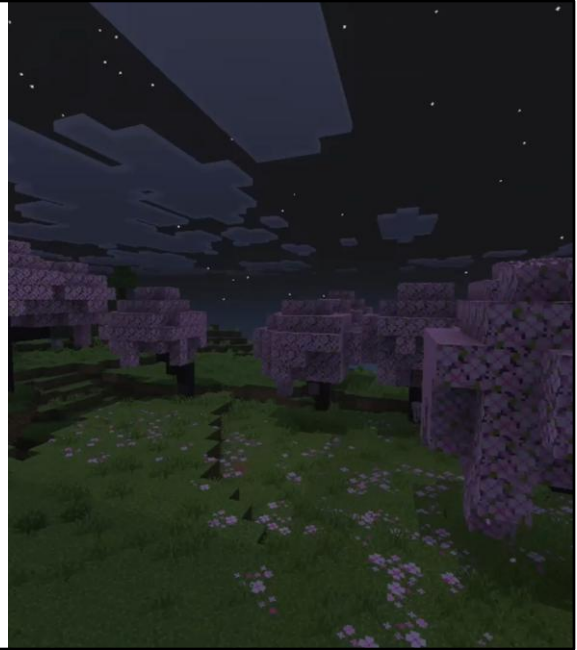
Redstone Circuitry: blocks emit light to indicate if a Redstone circuit is currently powered on or not

[GDC] Festival of Gaming

MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential



We need to keep in mind that lighting is already an integral part of Minecraft's game design. If we broke any of those existing expectations of light, then it would no longer feel like Minecraft.

We wanted to make sure we stayed physically based, but that didn't mean we couldn't put constraints on our system to keep it feeling "Minecrafty".

VIBRANT VISUALS

BRDF (Bidirectional Reflectance Distribution Function)

Started with Burley's implementation from Disney

Stuck with industry standard functions for pieces of the Cook-Torrance Reflection Model

Self-Shadowing function from Schlick-GGX, specifically the one in [UE4](#)

$$G(l, v, n) = G'(l, n)G'(v, n)$$
$$G'(x, n) = \frac{n \cdot x}{(n \cdot x)(1 - k) + k}$$
$$k = \frac{\alpha}{2}$$

Fresnel function from [Schlick](#)

$$F(v, h) = rf_0 + (1 - rf_0)(1 - (v \cdot h))^5$$

$rf_0 = \text{tinted index of refraction}$

Normal distribution function from [Trowbridge-Reitz/GGX](#)

$$D(h) = \frac{\alpha^2}{\pi((n \cdot h)^2(\alpha^2 - 1) + 1)^2}$$

VIBRANT VISUALS

Lighting Model

Direct

- 1 directional light
- Diffuse: Lambertian BRDF + Subsurface
- Specular: Cook-Torrance BRDF

Indirect Diffuse

- Fixed ambient
- Rebalanced Minecraft Sky + Block Light

Indirect Specular

- IBL (Sky Cubemap) + SSR
- Also employs the Cook-Torrance BRDF

Emissive



[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential

VIBRANT VISUALS

Shading Model

Do as much shading in Deferred as possible

Multi-Res Shading FTW

- Split up our deferred pass into multiple stages so that we can control resolution of each

Water and other transparencies are ForwardPBR

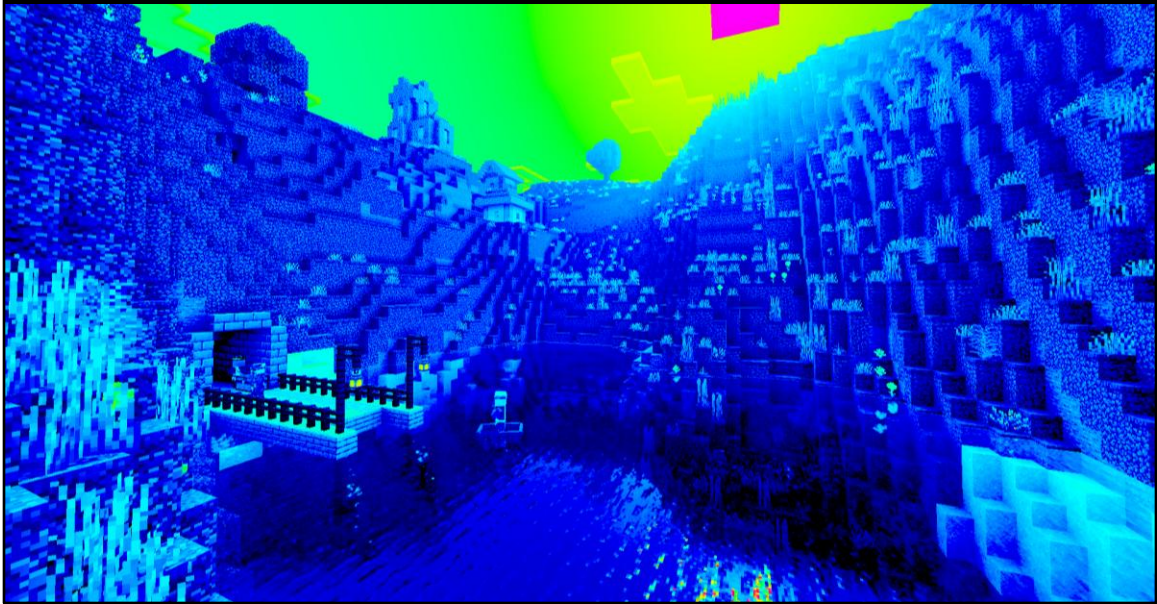
Balance is between how large the G-buffer is and how expensive lighting an individual pixel is

	Format	R	G	B	A
AlbedoMetalness	R8G8B8A8	Albedo			Metalness
NormalsMotionVectors	R16G16B16A16	Normals X,Y		Velocity dX,dY	
EmissiveAmbientRoughness	R16G16B16A16	Emissive	Block GI	Sky GI	Roughness

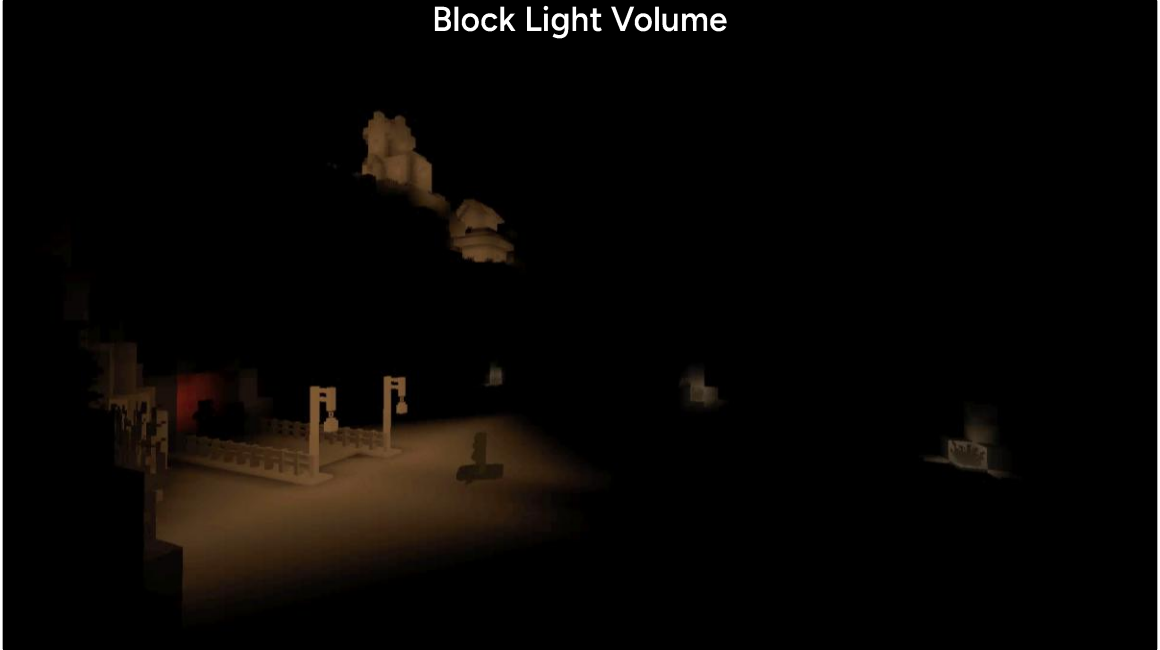
Final Shaded Scene



Luminance



Block Light Volume



Direct Diffuse



Direct Diffuse + Direct Specular



Indirect Specular



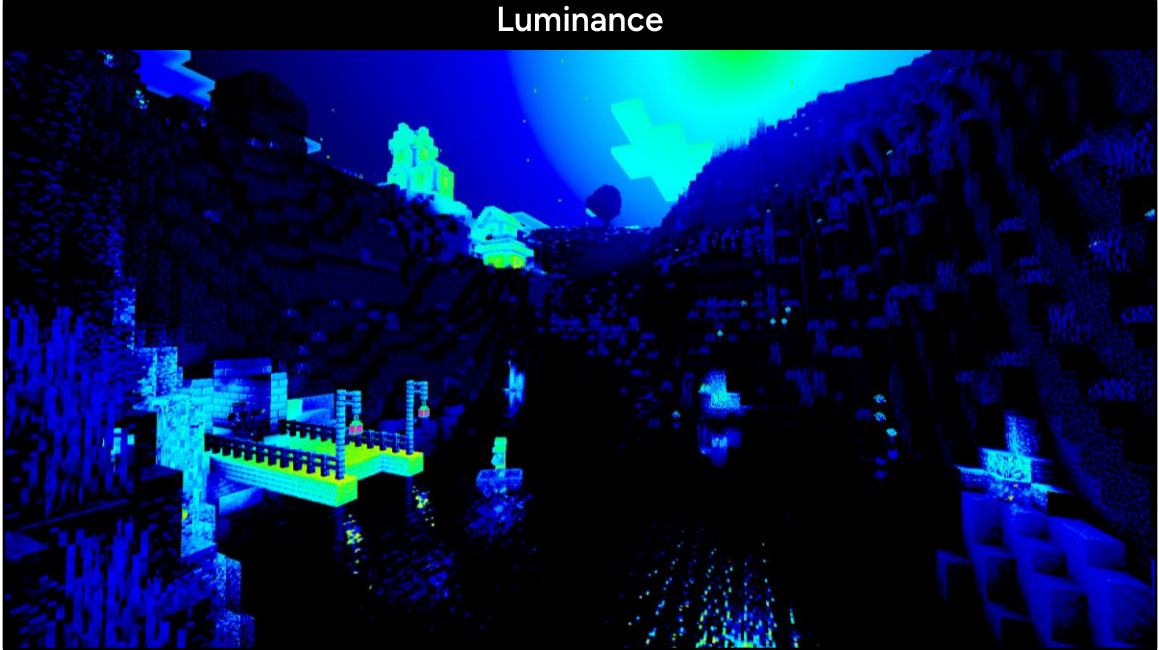
Final Shaded Scene



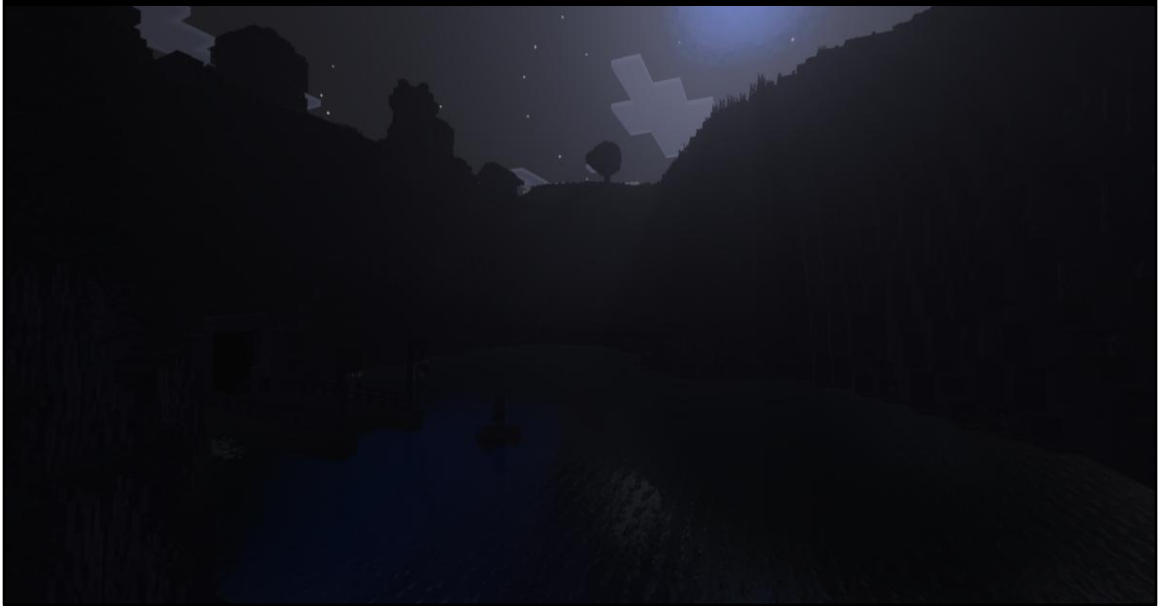
Final Shaded Scene



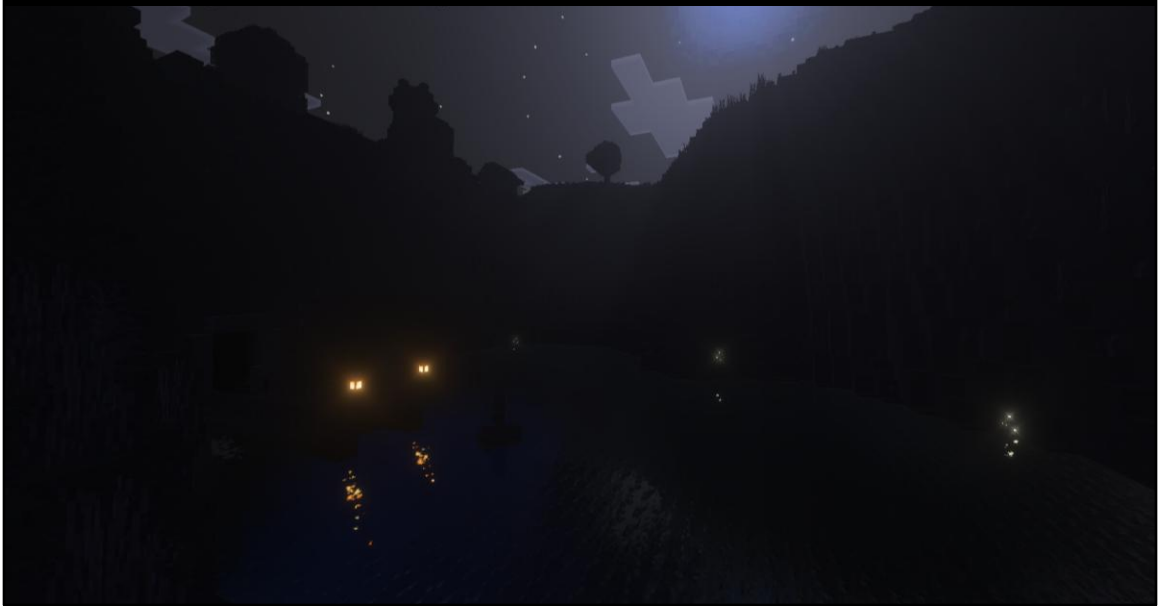
Luminance



Indirect Specular



Indirect Specular + Emissive





VIBRANT VISUALS

Sub-Surface Scattering Approximation

Modeling light that penetrates a surface

Important for Minecraft materials, e.g., slime, foliage, etc...

Algorithm Details

Calculating "transmission" depth

- Sample raw depth from shadow map occlusion test
- Compute distance between fragment and occluder
- Tuned to only allow for penetration up to 1 block

Direct diffuse term

- Energy conserving wrapped diffuse for $N * L$
- Transmitted diffuse with $-N * L$; no Fresnel component

Add transmitted diffuse term to direct diffuse scaled by transmission depth

 Festival of Gaming | MARCH 9-13, 2006
SAN FRANCISCO, CA

#GDC2026

We couldn't afford a dedicated BTDF, but we could do something directional in our diffuse term.

I could imagine a bunch of cases where this would fall apart with thin geometry, but we were able to put constraints on the thickness that keep subsurface leaking to a minimum.

VIBRANT VISUALS

Quantized Shadows and Reflections

Nothing revolutionary, just a little bit of math in the shader to quantize the world position used for those effects

```
vec3 getQuantizedWorldPosition(vec3 worldPos, vec3 viewPos, mat4 invView,
                               float texelGridSize, float normalPrecision) {
    vec3 approxSurfaceNormal = getApproxWorldSurfaceNormal(viewPos, invView);

    // Rounding the surface normal helps guard against artifacts that result
    // as imprecisions in the approximation of surface normal
    approxSurfaceNormal = normalize(round(approxSurfaceNormal /
        normalPrecision) * normalPrecision);

    // Quantize base world position
    vec3 worldPosRemainder = mod(worldPos, texelGridSize);

    // Remove anything in the normal direction from the quantized position
    worldPosRemainder -= dot(worldPosRemainder, approxSurfaceNormal) *
        approxSurfaceNormal;

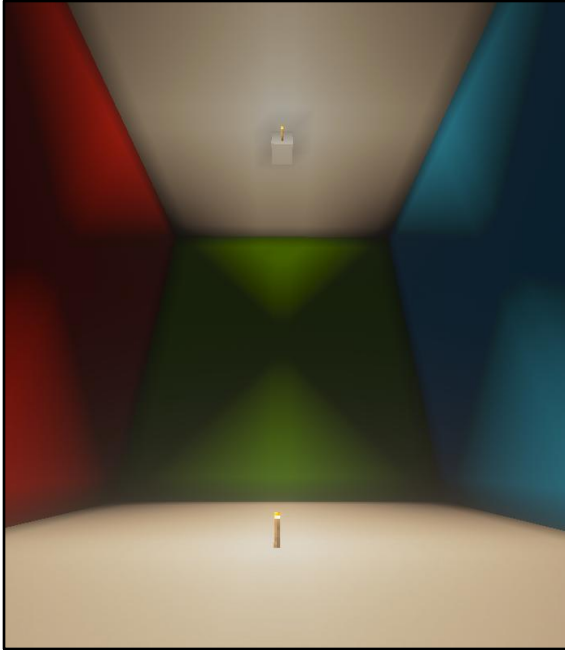
    return worldPos - worldPosRemainder;
}
```

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential





VIBRANT VISUALS

Lighting Calibration

We wanted to incorporate the traditional lighting terms in PBR, but how?

We asked ourselves: what is the luminous power of a torch?

Any guesses?

42 lumens?

1337 lumens?

1262.665039 lumens

>9000 lumens?

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

VIBRANT VISUALS

Lighting Calibration

1. Luminous power, lm , refers to the amount of light emitted over a given area
2. A torch has an intensity of 14 in Minecraft lighting units (MLU)
3. Minecraft light propagation follows cellular automata rules of Von Neumann Neighborhood $\rightarrow$ octahedron shape
4. MLU corresponds to the circumsphere radius of an octahedron:

$$r = 13.5; a = 19.091883$$

$$A = 2\sqrt{3}a^2 = 1262.665039 lm$$

5. Light attenuates at a linear rate in Minecraft, therefore:

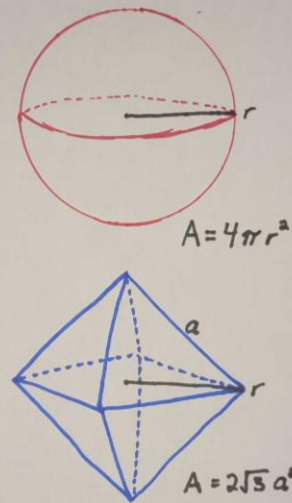
$$\frac{1262.665039}{14} = 90.190359$$

$$MLUtoLumens(x) = 90.190359x$$

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential



This is a little hand-wavey, but the important thing here is energy conservation. We know that Minecraft lighting doesn't behave exactly like lighting does in reality, but by understanding the core concepts, we can adapt our model. This really helped us reason with different relative levels of light between sources, and make the jump to PBR.

VIBRANT VISUALS

Colored Block Lighting

Extended traditional monochromatic light propagation to color

Instead of propagating RGB channels directly, propagate each type of light source

After propagation, colors for each type of light are accumulated into the final result

SWAR (SIMD-within-a-register)

Allows for execution of multiple math operations in a single instruction by dividing the bits of a type into "lanes"

E.g., lets us do 8 adds on 4-bit lanes in a single 32bit value

[GDC] Festival of Gaming | MARCH 9-13, 2006
SAN FRANCISCO, CA

#GDC2026

We're not propagating three times, once for each channel. But rather, we're propagating for each distinct type of light source, that's a unique combination of intensity and color. This approach lets us preserve colors even when multiple lights overlap and we can control the attenuation gradient per type of light. If we just did 3-channel RGB propagation, lights could suffer from hue-shifting, for instance an orange light turning red because the green channel is extinguished before the red channel.

VIBRANT VISUALS

Indirect Diffuse Sky Lighting

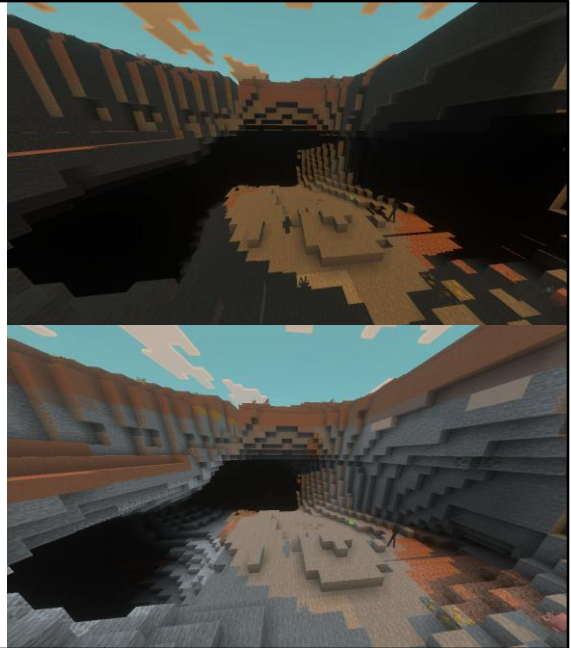
Free global illumination!? Yes, please!

Integration into lighting model

Vanilla sky light value, 0-15, is used as a measure of exposure to the sky

Normalized relative to the strength of the current directional light, either the Sun or Moon, for physical units

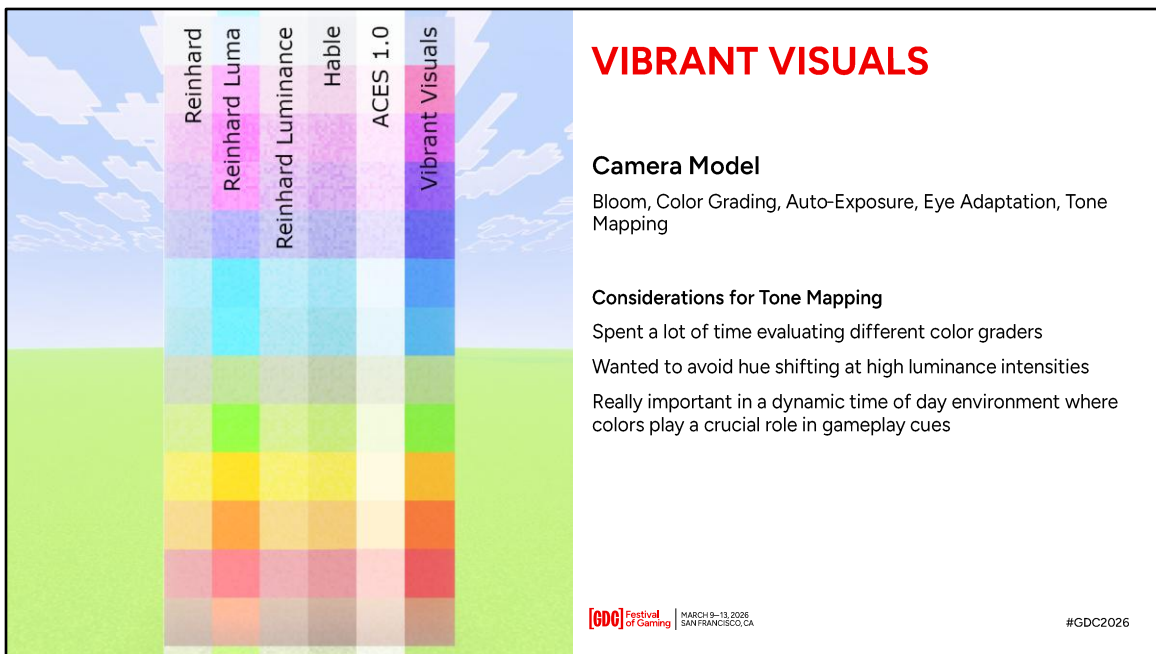
Great for approximating global illumination in cave entrances



[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential

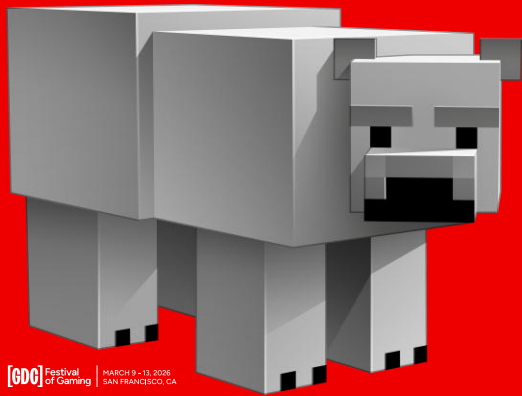


For the built-in Minecraft experience we felt it was important to avoid hue shifting or twisting at high luminance intensities, so we did our color grading in a color volume space rather than per channel.

The other tone mappers start to get washed out and hues start to shift.

Even at very high luminance values, our hues are preserved quite well. You can still tell the difference between the various blues and purples, and the oranges and yellows were retained as well. But as you can tell from the background, the scene is getting noticeably “brighter”.

PERFORMANCE TARGETS



Nitty-gritty details of end user hardware

[GDC] Festival of Gaming | MARCH 9 - 13, 2026
SAN FRANCISCO, CA
Classified as Microsoft Confidential

#GDC2026

PERFORMANCE TARGETS

Importance of Scalability

Each visual feature we built had to be scalable in terms of trading off perf for visual quality

Data-driving these sets of parameters allowed Dev and Tech Art to easily tune each feature for a given platform

Pre-defined presets: Performance and Visuals

We also allow for a Custom mode where people can turn on features that may potentially give them a bad experience in terms of frame rate, but we found that many people were willing to do this

[GDC] Festival
of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential



PERFORMANCE TARGETS

Framerate and Resolution of Vibrant Visuals

	Performance Mode		Visuals Mode			Performance Mode		Visuals Mode	
	FPS	Res	FPS	Res		FPS	Res	FPS	Res
Android – Tier 1	N/A	N/A	N/A	N/A	iOS – Tier 1	N/A	N/A	N/A	N/A
Android – Tier 2	N/A	N/A	N/A	N/A	iOS – Tier 2	60	480p	30	480p
Android – Tier 3	N/A	N/A	N/A	N/A	iOS – Tier 3	60	480p	30	720p
Android – Tier 4	30	480p	N/A	N/A	iOS – Tier 4	60	540p	30	720p
Android – Tier 5	30+	720p	30	720p	Console – Gen 8	30	1080p	30	1080p
Console – Gen 8.5	30	1080p	30	1440p	Console – Gen 9	60	4K	30	4K

PERFORMANCE TARGETS

iOS Device Tiering

Apple provides 2 values that are useful for deriving a device's tier: GPU Name and MTL GPU Family

We calculate a tier using a scoring formula:

1. Initialize to MTL GPU Family number + 2
 - Why add 2? So that the first chip, aka the A7, will work out to 1: $\text{Apple1} \Rightarrow (1 + 2) / 3 = 1$
2. +2 if the GPU name indicates an M-series SoC
3. +1 if the GPU name indicates a "Pro" variant
4. +2 if the GPU name is a "Max" or "Ultra" variant
5. Divide the final value by 3 and round down to the nearest whole number
 - Why divide by 3? GPU performance tends to improve noticeably every ~3 generations

We don't support Vibrant Visuals on Tier 1 due to lack of support for TextureCube Arrays

PERFORMANCE TARGETS

Android Device Tiering

Tier 1 **Adreno:** Anything before Adreno 500

Mali: Utgard arch or any GPU with the naming scheme Mali-4XX

PowerVR: Anything prior to the PowerVR 7000, including SGX gens

V3D/VideoCore: Anything prior to the V3D 4

Vivante: Any Vivante GPU

Tier 2 **Adreno:** Any Adreno 500 series GPU or later

Mali: Midgard arch or any GPU with the naming scheme Mali-T###

Nvidia Tegra: Tegra 1 through Tegra 4 or all Tegra K# series

PowerVR: All PowerVR 7000 or later

V3D/VideoCore: V3D 4 or later, including VideoCore VI

Tier 3 **Adreno:** Any Adreno 600 series GPU before the Adreno 640

Mali: Any Mali GPU based on the Bifrost arch

Nvidia Tegra: Tegra X series or later

PowerVR: PowerVR 9000 or later

V3D/VideoCore: V3D 7 or later

Tier 4 **Adreno:** Adreno 640 onwards or any Adreno 700 series

GPU **except** ray-tracing capable devices. Includes the Adreno 8cx, A11 and A21, but not the A32.

Mali: Valhall arch **except** ray-tracing capable devices (e.g., 4th gen/Mali-G615+)

Maleoon: Any Maleoon GPU which does not support ray-tracing

PowerVR: PowerVR A-Series or later (e.g. B-Series)

Xclipse: the Xclipse 530

Tier 5 **Adreno:** Adreno 740, Adreno X1, and Adreno A32 or later

Mali: Mali-G615/Mali-G715/Immortalis models or later

PowerVR: PowerVR C-Series or later (e.g. D-Series)

Xclipse: Xclipse 920 or later

We were careful to match these tiers to a bell curve based on our device telemetry

PERFORMANCE TARGETS

Final Considerations for Mobile

Android Graphics Dev

Really like developing for 4 different platforms

- Mali
- Exynos
- Adreno
- PowerVR

AGI is great, and vendors tend to have their own tools

Thermal Throttling

Factor of overall energy use

Multithreading or faster algorithms won't always address performance issues if energy use isn't considered

Thermal throttling will absolutely destroy your performance in a non-recoverable way and must be avoided at all costs

Consider keeping frame rates at 30fps to leave headroom on the device

Soak testing on and off battery can help identify these issues

Resolution

Probably the largest factor in mobile performance for deferred pipelines

Lots of phones have very high-resolution screens

Investment in upscaling is crucial if you want to hit safe, playable framerates

Frame graphs and different effect modules should be flexible in their resolution so that you can tune sub resolutions within the pipeline

COMMUNITY DRIVEN DEVELOPMENT



Getting feedback early and
often from our Creator
audiences

[GDC] Festival
of Gaming | MARCH 9 - 13, 2026
SAN FRANCISCO, CA

Classified as Microsoft Confidential

#GDC2026

One of Minecraft's greatest strengths is our community of Creators who like to modify the core game and build new experiences on top of it. You can script your own custom items or design your own cute mob or reskin literally every block in the game.

COMMUNITY DRIVEN DEVELOPMENT

We released well before we were done

Vibrant Visuals was available in Previews for roughly 2 years under the name of "Deferred Technical Preview"

Wanted to get feedback about the direction we were going

So, like, was it worth it?

More feedback than we could count

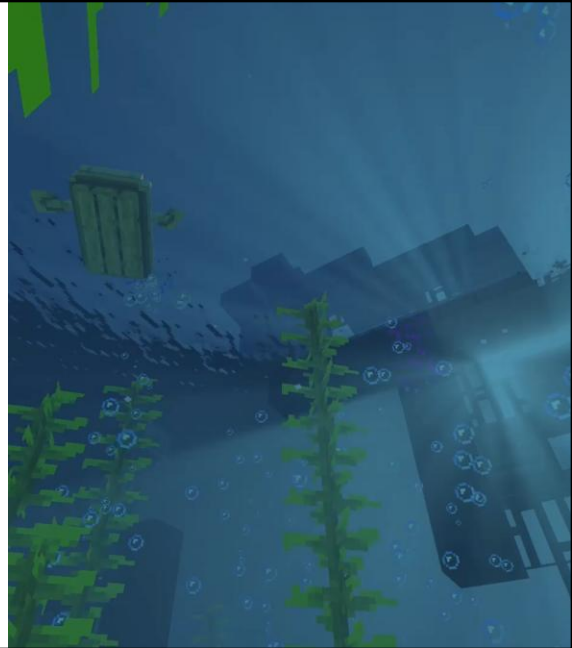
Allowed us to make a large change with a relatively small team

Creators had to bring their own "pack", but could customize the look and feel of their worlds via data-driving

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential





Different maps/packs by different Creators that are all built on the same Vibrant Visuals pipeline, but with customized textures and effect settings. Creators are really able to convey different moods and unique experiences. There's truly something for everyone.



COMMUNITY DRIVEN DEVELOPMENT

Develop in the open

Make use of "Early Access"

Encourage modding in your player community

Leverage your passionate players

Quickly validate new ideas

Often willing to overlook rough edges to be on bleeding edge

Tend to write detailed bug reports

Provide low barrier-to-entry modding tools and the community will build upon them

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

FUTURE

Where do we go from here?

- Always want to support the newest platforms
- Bringing Vibrant Visuals to Java
- Exploring even more advanced techniques
- Continuing to work closely with our community of Creators

[GDC] Festival
of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026



SPECIAL THANKS

Contributors

Aaron Heyse, Abhijit Zala, Adrien Givry, Andre Gracias, Andy Hill, Angelo Moyers, Anthony Cloudy, Aran Gilman, Brad Shuber, Bryan Yeo, Cameron Egbert, Chris Aduna, David Galloway, David Stout, Dayana Sharshukova, Dejan Dimic, Dylan Borg, Dom Arcamone, Emmaline Kelly, Eric Nguyen, Erik Davis, Ethan Koltz, Frank Xu, Gary Shaw, Grue Robinson, Hugo Machado, Isaac Calon, Isaac Dayton, Jaafer Sheriff, Jaakko Haapasalo, James Yarrow, Jasper Boerstra, Jessica Chen, Joule Garvin, Joey Montgomery, Jonathan Hoof, Joseph Cuen, Justin Tim, Karim Luccin, Kasia Swica, Kayla Kinnunen, Kris Munson, Li Yang, Maddie Psenka, Matt Mackowski, Matt Staubus, Micah Johnston, Michael Anderson, Michael Seydl, Michael Weilbacher, Mike Su, Mukund Agarwal, Oli Wright, Rey Brassard, Rod Reddekopp, Russell Gillette, Sam Hallam, Sandro Furini, Stacey Gray, Steve Anichini, Steve Avery, Steve Radomski, Teo Dutra, Tim Hagberg, Tim Stump, Todd Hayes, Tommaso Checchi, Torin Wiebelt, Yuhan Wu

[GDC] Festival of Gaming | MARCH 9-13, 2026
SAN FRANCISCO, CA

#GDC2026

Classified as Microsoft Confidential

Thanks

[GDC] Festival
of Gaming | MARCH 9 - 13, 2026
SAN FRANCISCO, CA

Classified as Microsoft Confidential

#GDC2026